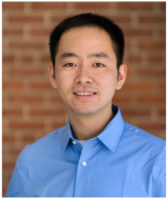
Communication-Efficient Decentralized Learning

Yuejie Chi

Carnegie Mellon University

EdgeComm Workshop, 2020

Acknowledgements



Boyue Li
CMU

Shicong Cen
CMU

Yuxin Chen
Princeton

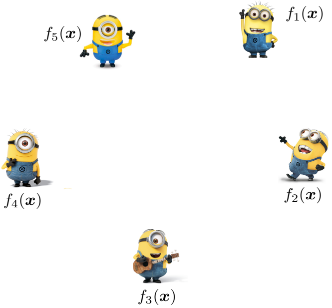
Communication-Efficient Distributed Optimization in Networks with Gradient Tracking and Variance Reduction, JMLR, 2020.

Distributed empirical risk minimization

Distributed/Federated learning: due to privacy and scalability, data are distributed at multiple locations / workers / agents.

Let $\mathcal{M} = \cup_i \mathcal{M}_i$ be a data partition with equal splitting:

$$f(\boldsymbol{x}) := \frac{1}{n} \sum_{i=1}^{n} f_i(\boldsymbol{x}), \quad \text{where} \quad f_i(\boldsymbol{x}) := \frac{1}{(N/n)} \sum_{\boldsymbol{z} \in \mathcal{M}_i} \ell(\boldsymbol{x}; \boldsymbol{z}).$$



N = number of total samples

n = number of agents

$\underbrace{N/n}_{m}$ = number of local samples

Decentralized ERM - algorithmic framework

$$\text{minimize}_{\boldsymbol{x}} \quad f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

$$\Downarrow$$

$$\text{minimize}_{\boldsymbol{x}_i} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}_i) \quad \text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{x}_j$$

Decentralized ERM - algorithmic framework

$$\text{minimize}_{\boldsymbol{x}} \quad f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

$$\Downarrow$$

$$\text{minimize}_{\boldsymbol{x}_i} \quad \frac{1}{n}\sum_{i=1}^{n} \boxed{f_i(\boldsymbol{x}_i)} \quad \text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{x}_j$$

- **Local computation:** agents update local estimate;

$\Rightarrow$ need to be scalable!

Decentralized ERM - algorithmic framework

$$\text{minimize}_{\boldsymbol{x}} \quad f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

$$\Downarrow$$

$$\text{minimize}_{\boldsymbol{x}_i} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}_i) \quad \text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{x}_j$$

- **Local computation:** agents update local estimate;

  $\Rightarrow$ need to be scalable!

- **Global communications:** agents exchange for consensus;

  $\Rightarrow$ need to be communication-efficient!

Decentralized ERM - algorithmic framework

$$\text{minimize}_{\boldsymbol{x}} \quad f(\boldsymbol{x}) := \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$$

$$\Downarrow$$

$$\text{minimize}_{\boldsymbol{x}_i} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x}_i) \quad \text{subject to} \quad \boldsymbol{x}_i = \boldsymbol{x}_j$$

- **Local computation:** agents update local estimate;

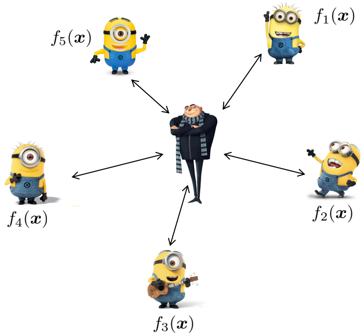
  $\Rightarrow$ need to be scalable!

- **Global communications:** agents exchange for consensus;

  $\Rightarrow$ need to be communication-efficient!

Guiding principle: more local computation leads to less communication

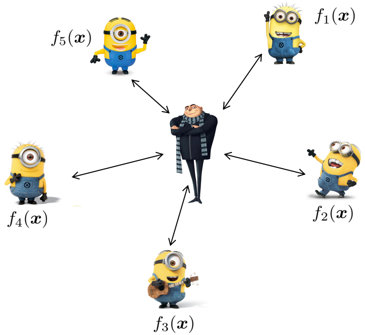
Two distributed schemes



Master/slave model

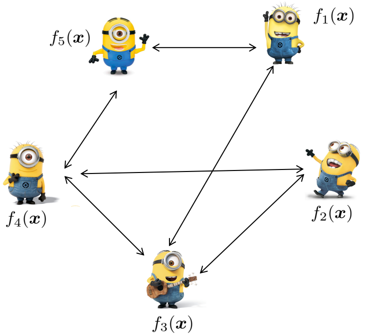
PS coordinates <u>global</u> information sharing

Two distributed schemes



Master/slave model

PS coordinates global information sharing



Network model

agents share local information over a graph topology

Distributed first-order methods in the master/slave setting

local data

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \nabla f(\boldsymbol{x}^t),\ \boldsymbol{x}^t)$$

$$\boldsymbol{x}^t = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{t-1}$$

parameter consensus

$$\nabla f(\boldsymbol{x}^t) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^t)$$

gradient consensus

Distributed first-order methods in the master/slave setting

local data

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \nabla f(\boldsymbol{x}^t),\ \boldsymbol{x}^t)$$

$$\boldsymbol{x}^t = \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i^{t-1}$$

parameter consensus

$$\nabla f(\boldsymbol{x}^t) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(\boldsymbol{x}^t)$$

gradient consensus

Distributed first-order methods in the master/slave setting

local data

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \nabla f(\boldsymbol{x}^t),\ \boldsymbol{x}^t)$$

$$\boldsymbol{x}^t = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{t-1}$$

parameter consensus

$$\nabla f(\boldsymbol{x}^t) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^t)$$

gradient consensus

Distributed first-order methods in the master/slave setting

local data

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \nabla f(\boldsymbol{x}^t),\ \boldsymbol{x}^t)$$

$$\boldsymbol{x}^t = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{t-1}$$

parameter consensus

$$\nabla f(\boldsymbol{x}^t) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^t)$$

gradient consensus

Distributed first-order methods in the master/slave setting

local data

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \nabla f(\boldsymbol{x}^t),\ \boldsymbol{x}^t)$$

$$\boldsymbol{x}^t = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{t-1}$$

parameter consensus

$$\nabla f(\boldsymbol{x}^t) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^t)$$

gradient consensus

Distributed Approximate NEwton (DANE) (Shamir et. al., 2014):

$$\boldsymbol{x}_i^t = \operatorname*{argmin}_{\boldsymbol{x}}\ f_i(\boldsymbol{x}) - \left\langle \nabla f_i(\boldsymbol{x}^{t-1}) - \nabla f(\boldsymbol{x}^{t-1}), \boldsymbol{x} \right\rangle + \frac{\mu}{2}\left\|\boldsymbol{x} - \boldsymbol{x}^{t-1}\right\|_2^2$$

- Quasi-Newton method and less sensitive to ill-conditioning.

Distributed first-order methods in the master/slave setting

local data

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \nabla f(\boldsymbol{x}^t),\ \boldsymbol{x}^t)$$

$$\boldsymbol{x}^t = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i^{t-1}$$

parameter consensus

$$\nabla f(\boldsymbol{x}^t) = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\boldsymbol{x}^t)$$

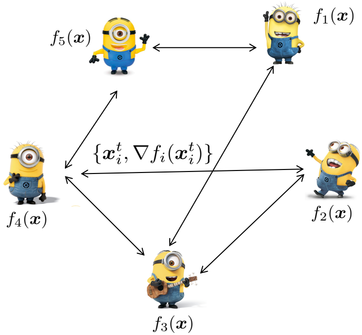
gradient consensus

Distributed Stochastic Variance-Reduced Gradients (Cen et. al., 2020):

$$\boldsymbol{x}_i^{t,s} \Longleftarrow \boldsymbol{x}_i^{t,s-1} - \eta \underbrace{\boldsymbol{v}_i^{t,s-1}}_{\text{variance-reduced stochastic gradient}}, \qquad s = 1, 2, \dots,$$

- Better local computation efficiency.

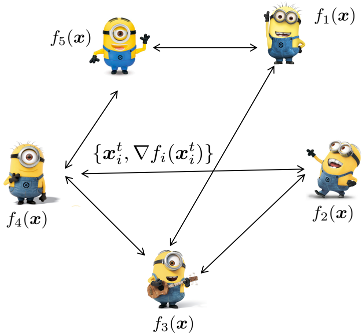
Naive extension to the network setting

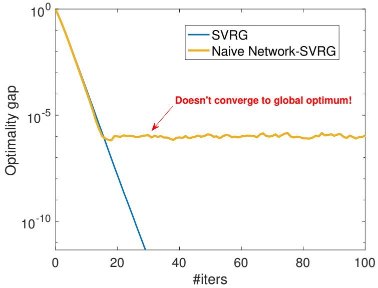


- **Communicate:** agent transmits $\{\boldsymbol{x}_i^t, \nabla f_i(\boldsymbol{x}_i^t)\}$;
- **Compute:**

$$\boldsymbol{x}_i^t \Leftarrow \texttt{LocalUpdate}\big(f_i, \underbrace{\mathsf{Avg}\{\nabla f_j(\boldsymbol{x}_j^t)\}_{j\in\mathcal{N}_i}}_{\text{surrogate of } \nabla f(\boldsymbol{x}^t)}, \underbrace{\mathsf{Avg}\{\boldsymbol{x}_j^t\}_{j\in\mathcal{N}_i}}_{\text{surrogate of } \boldsymbol{x}^t}\big)$$

Naive extension to the network setting

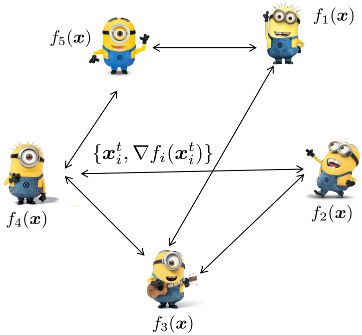


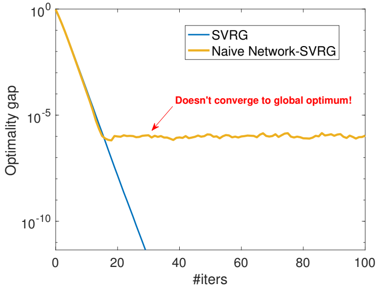


- **Communicate:** agent transmits $\{\boldsymbol{x}_i^t, \nabla f_i(\boldsymbol{x}_i^t)\}$;
- **Compute:**

$$\boldsymbol{x}_i^t \Leftarrow \texttt{LocalUpdate}(f_i, \underbrace{\mathsf{Avg}\{\nabla f_j(\boldsymbol{x}_j^t)\}_{j\in\mathcal{N}_i}}_{\text{surrogate of } \nabla f(\boldsymbol{x}^t)}, \underbrace{\mathsf{Avg}\{\boldsymbol{x}_j^t\}_{j\in\mathcal{N}_i}}_{\text{surrogate of } \boldsymbol{x}^t})$$

Naive extension to the network setting





- **Communicate:** agent transmits $\{\boldsymbol{x}_i^t, \nabla f_i(\boldsymbol{x}_i^t)\}$;
- **Compute:**

$$\boldsymbol{x}_i^t \Leftarrow \texttt{LocalUpdate}(f_i, \underbrace{\text{Avg}\{\nabla f_j(\boldsymbol{x}_j^t)\}_{j\in\mathcal{N}_i}}_{\text{surrogate of } \nabla f(\boldsymbol{x}^t)}, \underbrace{\text{Avg}\{\boldsymbol{x}_j^t\}_{j\in\mathcal{N}_i}}_{\text{surrogate of } \boldsymbol{x}^t})$$

Consensus needs to be designed carefully in the network setting!

Average dynamic consensus

Assume that each agent generates some time-varying quantity r_j^t.

How to track its the dynamic average $\frac{1}{n}\sum_{j=1}^{n} r_j^t = \frac{1}{n}\mathbf{1}_n^\top \boldsymbol{r}^t$ in each of the agents, where $\boldsymbol{r}^t = [r_1^t, \cdots, r_n^t]^\top$?

Average dynamic consensus

Assume that each agent generates some time-varying quantity r_j^t.

How to track its the dynamic average $\frac{1}{n}\sum_{j=1}^{n} r_j^t = \frac{1}{n}\mathbf{1}_n^\top \boldsymbol{r}^t$ in each of the agents, where $\boldsymbol{r}^t = [r_1^t, \cdots, r_n^t]^\top$?

- Dynamic average consensus (Zhu and Martinez, 2010):

$$\boldsymbol{q}^t = \underbrace{\boldsymbol{W}\boldsymbol{q}^{t-1}}_{\text{mixing}} + \underbrace{\boldsymbol{r}^t - \boldsymbol{r}^{t-1}}_{\text{correction}},$$

where $\boldsymbol{q}^t = [q_1^t, \cdots, q_n^t]^\top$ and $\boldsymbol{W}$ is the mixing matrix.

Average dynamic consensus

Assume that each agent generates some time-varying quantity r_j^t.

How to track its the dynamic average $\frac{1}{n}\sum_{j=1}^{n} r_j^t = \frac{1}{n}\mathbf{1}_n^\top \boldsymbol{r}^t$ in each of the agents, where $\boldsymbol{r}^t = [r_1^t, \cdots, r_n^t]^\top$?

- Dynamic average consensus (Zhu and Martinez, 2010):

$$\boldsymbol{q}^t = \underbrace{\boldsymbol{W}\boldsymbol{q}^{t-1}}_{\text{mixing}} + \underbrace{\boldsymbol{r}^t - \boldsymbol{r}^{t-1}}_{\text{correction}},$$

  where $\boldsymbol{q}^t = [q_1^t, \cdots, q_n^t]^\top$ and $\boldsymbol{W}$ is the mixing matrix.

- **Key property:** the average of $\{q_i^t\}$ dynamically tracks the average of $\{r_i^t\}$;

$$\mathbf{1}_n^\top \boldsymbol{q}^t = \mathbf{1}_n^\top \boldsymbol{r}^t,$$

M. Zhu and S. Martnez. "Discrete-time dynamic average consensus." Automatica 2010.

Gradient tracking

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \cancel{\nabla f(\boldsymbol{x}^t)} \xrightarrow{} \boldsymbol{s}_j^t,\ \cancel{\boldsymbol{x}^t} \xrightarrow{} \boldsymbol{y}^t)$$

- Parameter averaging:

$$\boldsymbol{y}_j^t = \sum\nolimits_{k \in \mathcal{N}_j} w_{jk} \boldsymbol{x}_k^{t-1},$$

- Gradient tracking:

$$\boldsymbol{s}_j^t = \sum\nolimits_{k \in \mathcal{N}_j} w_{jk} \boldsymbol{s}_k^{t-1} + \underbrace{\nabla f_j(\boldsymbol{y}_j^t) - \nabla f_j(\boldsymbol{y}_j^{t-1})}_{\text{gradient tracking}}.$$

Gradient tracking

$$\boldsymbol{x}_i^t \Longleftarrow \texttt{LocalUpdate}(f_i,\ \cancelto{\boldsymbol{s}_j^t}{\nabla f(\boldsymbol{x}^t)},\ \cancelto{\boldsymbol{y}^t}{\boldsymbol{x}^t})$$

- Parameter averaging:

$$\boldsymbol{y}_j^t = \sum\nolimits_{k \in \mathcal{N}_j} w_{jk} \boldsymbol{x}_k^{t-1},$$

- Gradient tracking:

$$\boldsymbol{s}_j^t = \sum\nolimits_{k \in \mathcal{N}_j} w_{jk} \boldsymbol{s}_k^{t-1} + \underbrace{\nabla f_j(\boldsymbol{y}_j^t) - \nabla f_j(\boldsymbol{y}_j^{t-1})}_{\text{gradient tracking}}.$$

We can now apply the same DANE and SVRG-type local updates!

Linear Regression: Well-Conditioned

$$f_i(\boldsymbol{x}) = \|\boldsymbol{y}_i - \boldsymbol{A}_i\boldsymbol{x}\|_2^2, \quad \boldsymbol{A}_i \in \mathbb{R}^{1000\times 40}$$

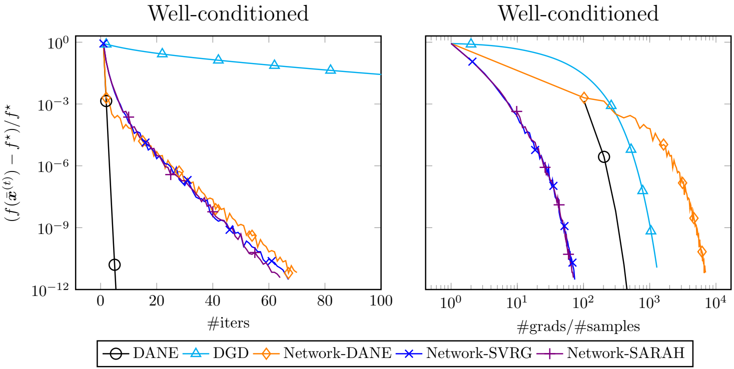


Figure: The optimality gap w.r.t. iterations and gradients evaluation. The condition number $\kappa = 10$. ER graph ($p = 0.3$), 20 agents.

Linear Regression: Ill-Conditioned

$$f_i(\boldsymbol{x}) = \|\boldsymbol{y}_i - \boldsymbol{A}_i\boldsymbol{x}\|_2^2, \quad \boldsymbol{A}_i \in \mathbb{R}^{1000\times 40}$$

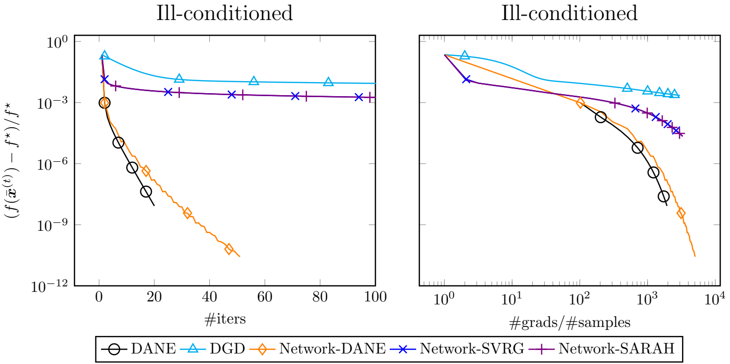
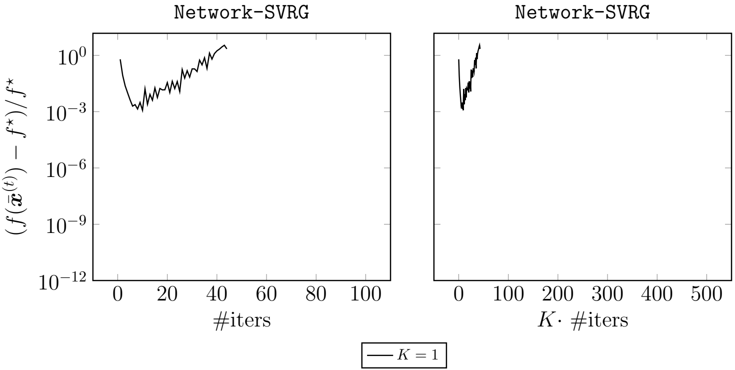


Figure: The optimality gap w.r.t. iterations and gradients evaluations. The condition number $\kappa = 10^4$. ER graph ($p = 0.3$), 20 agents.

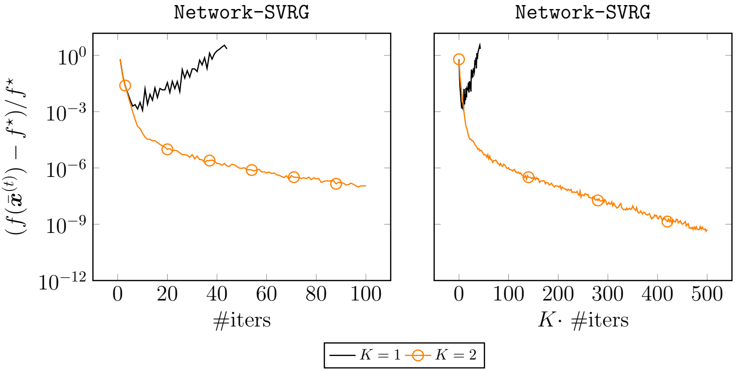
Extra Mixing

The mixing rate of the graph $\alpha_0 = 0.922$. A single round of mixing within each iteration cannot ensure the convergence of `Network-SVRG`.



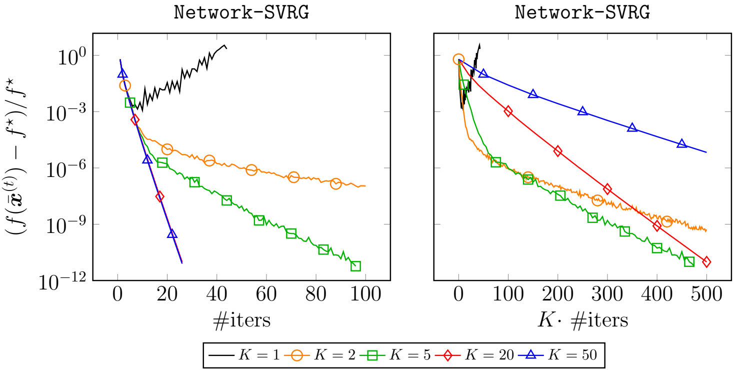
Extra Mixing

The mixing rate of the graph $\alpha_0 = 0.922$. A single round of mixing within each iteration cannot ensure the convergence of `Network-SVRG`.



Extra Mixing

The mixing rate of the graph $\alpha_0 = 0.922$. A single round of mixing within each iteration cannot ensure the convergence of `Network-SVRG`.



Final remarks

- gradient tracking provides a way to extend master/slave algorithms (DANE and SVRG) to network settings;
- probing computational-communication trade-offs by employing different local updates and extra mixing;

Future work:

- convergence analysis in the nonconvex case.

Thank you!

Communication-Efficient Distributed Optimization in Networks with Gradient Tracking
B. Li, S. Cen, Y. Chen, and Y. Chi, JMLR 2020.